

AMENDMENTS TO THE SPECIFICATION

Please amend the Specification by replacing the text starting at page 18, line 13, to page 30, line 20 with the following:

```
public class SKU {  
  
    protected String productName;  
  
    protected Double price;  
  
    protected String category;  
  
    public SKU(String productName, Double price) {  
  
        this.productName = productName;  
  
        this.price = price; }  
  
    public SKU(String productName, Double price, String category) {  
  
        this(productName, price);  
  
        this.category = category; }  
  
    public String getProductName() {return productName;}  
  
    public Double getPrice() {return price;}  
  
    public String getCategory() { return category;}  
  
    public String toString() {  
  
        return productName + "(price=" + price + "category=" + category + ");I  } }
```

By way of example, the following is the Java programmable code for a DiscountGroup class:

```
public class DiscountGroup {  
  
    Vector discounts = new Vector();
```

```
Double discountTotal = new Double(0.0);

public DiscountGroup() { }

public void addDiscount(Discount discount) {

    discounts add(discount);

    double dct = discountTotal.doubleValue();

    dct += discount.getDiscountValue().doubleValue();

    discountTotal = new Double (dct); }

public Boolean containsDiscount(Discount discount) {

    if (discounts contains(discount)) {

        return new Boolean(true);

    } else {

        return new Boolean(false); } }

public Boolean containsItem(OrderEntry oe) {

    for (int i=0; i < discounts size(); i++) {

        Discount discount = (Discount)discounts get(i);

        if (discount containsItem(oe).booleanValue () == true) return new Boolean(true);}

    return new Boolean(false); }

public Boolean containsItems(Discount discount) {

    Vector orderEntryItems = discount.getItems();

    for (int i=0; i < orderEntryItems.size(); i++) {

        OrderEntry oe = (OrderEntry)orderEntryItems.get(i);

        if(containsItem(oe).booleanValue() == true) return new Boolean(true);}

    return new Boolean(false); // didn't find any items }
```

```
public String toString(){  
  
    return "discountGroup=(discountTotal=" + discountTotal + "discounts="+ discounts +")"; } }
```

By way of example, the following is the Java programmable code for an order class:

```
public class Order {  
  
    static int nextId = 0;  
  
    protected Double orderId;  
  
    protected Vector orderEntries = new Vector();  
  
    public Order() {  
  
        orderId new Double(nextId++); }  
  
    public Vector getOrderEntries() {  
  
        return orderEntries; }  
  
    public void addOrderEntry(OrderEntry oe) {  
  
        orderEntries.add(oe); }  
  
    public Double getOrderId() {  
  
        return ordered ; }  
  
    public void selectDiscountGroup(Vector discounts) { }  
  
    public String toString() {  
  
        return "order=(orderId=" + orderId + "orderEntries="+ orderEntries +")"; } }
```

By way of example, the following is the Java programmable code for an OrderEntry class:

```
public class OrderEntry {  
  
    static int nextId = 0  
  
    protected Double orderEntryId ; //order entry id  
  
    protected Double orderId; // part of
```

```
protected SKU product ;

protected String productName ;

protected Double quantity ;

protected Double totalPrice ; // quantity * price

public OrderEntry() {

    orderEntryId = new Double(nextId++) ; }

public OrderEntry(String productName Double quantity) {

    this();

    this.productName = productName ;

    this.quantity = quantity ; }

public OrderEntry(SKU product. Double quantity) {

    this();

    this.product = product;

    this.productName = product.getProductName();

    this.quantity = quantity ;

    double tp = quantity.doubleValue() * product.getPrice().doubleValue();

    totalPrice = new Double(tp) ; }

public Double getOrderId() {

    return orderId ; }

public void setOrderId(Double orderId) {

    thisorderId = orderId ; }

public Double getOrderEntryId() {

    return orderEntryId ; }
```

```
public SKU getProduct() { return product ;}

public void setProduct(SKU product) {

    this.product = product;

    this.productName = product.getProductName(); }

public Double getPrice() { return product.getPrice(); }

public Double getTotalPrice() { return totalPrice ; }

public String getProductName() { return productName; }

public Double getQuantity() {return quantity; }

public void setQuantity(Double quantity) {

    this.quantity = quantity ;

    double tp = quantity.doubleValue() * product.getPrice().doubleValue();

    totalPrice = new Double(tp) ; }

public String toString() {

return "orderEntry=(id =" + orderEntryId + "orderId=" + orderId + "product="+

    productName + "quantity="+ quantity + ")"; } }
```

By way of example, the following is the Java programmable code for a Discount class:

```
public class Discount {

String name ="" ; // label used for identification

String type = "percentage" ; // default ???

Double factor1 = new Double(10.0) ; // defaults ???

Double factor2 = new Double(10.0) ; // defaults ???

Double discountValue = new Double(0.0) ;

Vector items = new Vector() ; // items to which this discount has been applied
```

```
public Discount() { }

public Discount(String type, Double factor1) {

    this.type = type ;

    this.factor1 = factor1 ;}

public Double applyDiscount(OrderEntry oe) {

    double discount = 0.0;

    double fact1 = factor1.doubleValue() ;

    double price = oe.getPrice().doubleValue();

    if (type.equals("percentageoff")) {

        discount = fact1 * price; // factor1 is percentage

    } else if (type.equals("free-merchandise")) {

        discount = fact1 * price ; // factor1 = N items

    } else if (type.equals("fixed-amount")) {

        discount = fact1 ; // factor1 = dollars}

    discountValue = new Double(discount) ;

    items.add(oe) ; //add item

    return discountValue ; }

public Double applyDiscount(OrderEntry oe1, OrderEntry oe2) {

    double discount = 0.0 ;

    double fact1 = factor1.doubleValue() ;

    double price1 = oe1.getPrice().doubleValue() ;

    double price2 = oe2.getPrice().doubleValue() ;

    if (type.equals("percentage-off")) {
```

```
        discount= fact1 * (price1 + price2) ; //factor 1 is percentage
    } else if (type.equals("free-merchandise")) {
        discount = fact 1 * (price1 + price2) ; //factor 1 = N items
    } else if(type.equals("fixed-amount")) {
        discount = fact1; //factor1 = dollars}
    discountValue = new Double(discount) ;
    items.add(oe1) ;
    items.add(oe2) ;
    return discountValue ; }

public String getName() { return name; }

public void setName (String name) {
    this.name = name; }

public void setType(String type) {
    this.type = type ; }

public void setFactor1(Double factor1) {
    this.factor1 = factor1 ; }

public Double appyDiscount(Order o) {
    return new Double (0.0) ; }

public double getdiscountValue() { return discountValue; }

public Boolean containsItem(OrderEntry oe) {
    if (items.contains(oe)) {
        return new Boolean (true);
    } else {
```

```

        return new Boolean (false) ;    } }

public Boolean containsItems(Vector orderEntryItems) {

    for(int i=0 ; i< orderEntryItems.size(); i++) {

        OrderEntry oe= (OrderEntry)orderEntryItems.get(i);

        if(containsItem(oe).booleanValue() == false) return new Boolean (false) ;

        return new Boolean (true) ; // found all items  }

public Vector getItems() { return items ; }

public String toString() {

    return name + "=(type = " + type + "factor1 + "discount=" + discountValue +")" ; }}

```

By way of example, the following is the Java programmable code for a Discount RuleSet class:

```

RuleSet WCSDiscount1 (

    InferenceMethod(Forward2)

    Types (Order : test.Order)

    Types (OrderEntry: test.OrderEntry)

    Types(SKU: test.SKU)

    Types (Discount: test.Discount)

    Types (DiscountGroup: test.DiscountGroup)

    Types(myString : java.lang.String)

    Import(com.ibm.able.beans.rules.AbleWorkingMemoryLib)

    Variables(

        productA SKU ("A", 100.0) // Name, price

        productB SKU ("B", 100.0) // Name, price

```



```
productC SKU ("C", 100.0) // Name, price

order Order ()

orderEntry 1 Order Entry ()

orderEntry 2 Order Entry ()

orderEntry 3 Order Entry ()

newDiscount Discount ()

newDiscountGroup DiscountGroup()

bestGroup DiscountGroup()

phase myString ("0") // turns rules on/off

falseValue Boolean(false)

trueValue Boolean(true)

rc List() )

InputVars()

OutputVars()

Rules init(

    : orderEntry1.product = productA

    : orderEntry1.quantity = 1

    : orderEntry1.orderId = 0

    : orderEntry2.product = productB

    : orderEntry2.quantity = 1

    : orderEntry2.orderId = 0

    : orderEntry3.product = productC

    : orderEntry3.quantity = 1
```

```

: orderEntry3.orderId = 0

: rc=invoke(order, "addOrderEntry", orderEntry1) ;

: rc=invoke(order, "addOrderEntry", orderEntry2) ;

: rc=invoke(order, "addOrderEntry", orderEntry3) ;

: rc=assert(wm, phase)

: rc=assert(wm, order)

: rc=assert(wm, orderEntry 1)

: rc=assert(wm, orderEntry 2)

: rc=assert(wm, orderEntry3)

: rc=assert(wm, productA)

: rc=assert(wm, productB)

: rc=assert(wm, productC) )

```

Rules (

```

Rule_Discount_1 [4]: when ( phase myString ( phase="0") &
    item OrderEntry (item.orderId = order.orderId),
    item.productName = "A" ) &
    discount Discount !(discount.name = "A") )

```

```

do (
    newDiscount = createInstance("test.Discount")
    rc= newDiscount.name "A"
    rc= newDiscount.type = "free-merchandise"
    rc= newDiscount.factor1 = 1 // buy A, get 1 A free
    rc=invoke(newDiscount, "applyDiscount", item)

```

```
rc=assert(wm, newDiscount)

newDiscountGroup = createInstance("test.DiscountGroup")

rc=invoke(newDiscountGroup, "addDiscount", newDiscount)

rc=assert(wm, newDiscountGroup)    )
```

Rule_Discount_3 [4]: when (phase myString (phase= "0") &

```
    item OrderEntry (item.orderId = order.orderId,
                    item.productName = "C") &
    discount Discount !(discount.name "C")    )
```

do (

```
    newDiscount = createInstance("test. Discount")

    rc= newDiscount.name = "C"

    rc= newDiscount.type = "percentage-off"

    rc= newDiscount.factor1 = 0.10 // buy C get 10% off

    rc=invoke(newDiscount, "applyDiscount", item)

    rc=assert(wm, newDiscount)

    newDiscountGroup = createInstance("test.DiscountGroup")

    rc=invoke(newDiscountGroup, "addDiscount", newDiscount)

    rc=assert(wm, newDiscountGroup)    )
```

Rule_Discount_2 [4]: when (phase myString (phase= "0") &

```
    item OrderEntry (item.orderId = order.orderId),
    item.productName = "A") &
    item2 OrderEntry (item2.orderId = order.orderId),
    item2.productName = "B" ) &
```

```

        discount Discount !(discount.name = "A+B")
    )

do (

    newDiscount = createInstance("test. Discount")

    rc= newDiscount.name = "A+B"

    rc= newDiscount.type = "percentage-off"

    rc= newDiscount.factor1 = 0.05 // buy A+B get 5% off both

    rc=invoke(newDiscount, "applyDiscount", item, item2)

    rc=assert(wm, newDiscount)

    newDiscountGroup = createInstance("test.DiscountGroup")

    rc=invoke(newDiscountGroup, "addDiscount", newDiscount)

    rc=assert(wm, newDiscountGroup)
)

Rule_ Discount_ 4 [4]: when ( phase myString ( phase="0") &

    item OrderEntry(item.orderId = order.orderId),

    item.productName = "B") &

    item2 OrderEntry (item2.orderId = order.orderId),

    item2.productName = "C") &

    discount Discount !(discount.name= "B+C")
)

do (

    newDiscount = createInstance("test.Discount")

    rc= newDiscount.name = "B+C"

    rc= newDiscount.type = "fixed-amount"

    rc= newDiscount.factor1 = 10.00 // buy B+C get $10 off

    rc=invoke(newDiscount, "applyDiscount", item, item2)

```

```
rc=assert(wm, newDiscount)

newDiscountGroup = createInstance("test.DiscountGroup")

rc=invoke(newDiscountGroup, "addDiscount", newDiscount)

rc=assert(wm, newDiscountGroup)    )
```

Rule_StartPhase1 [1]: when (phase myString (phase= "0"))

```
do (

    rc=retract(wm, phase)

    phase = "1"

    rc=assert(wm, phase)    )
```

Rule_ 6 [4]: when (phase myString (phase= "1") &

```
    discount Discount (discount.discountValue > 0.0) &

    discountGroup DiscountGroup (falseValue=invoke(discountGroup,

"containsDiscount", discount),

                                falseValue=invoke(discountGroup, "containsItems", discount)) )

do (

    rc=invoke(discountGroup, "addDiscount", discount)    )
```

Rule_StartPhase2 [1]: when (phase myString (phase= "1"))

```
do (

    rc= retract(wm, phase)

    phase = "2"

    rc=assert(wm, phase)    ) )
```